MANAGING SOFTWARE REQUIREMENTS

MAJ Joseph P. Dupont and
MAJ Robert W. Cummins Jr.

Introduction

The critical importance of software in today's high-tech world cannot be underestimated. As such, software development teams must be mindful of developing quality software on time, within budgets, and capable of meeting the customer's real needs. Despite this, however, a study by the Standish Group in 1994 reported, "A staggering 31 percent of projects will be canceled before they ever get completed, 52.7 percent of projects will cost 189 percent of their original estimates and require 222 percent more time than originally estimated, and only about 16 percent of software projects are completed on time and on budget."

The Army is certainly not immune from these results and sometimes is even more susceptible. One reason may be that DOD systems tend to be more complex and have more diverse requirements. In a recent Army project failure, the original project estimate was 18-24 months for completion at a cost of $22 million. The reality was 4 years, $70-$110 million, and the project was canceled with no delivery. The program manager (PM) noted the following as some of the reasons for the failure:

- Extremely high defect count with poor resolution trend,
- Negative combat developer feedback on system performance,
- Underestimated magnitude of the work, and
- Inadequate monitoring of daily and weekly requirements.

Inadequate requirements determination is often the cause of these abysmal results. Requirement errors are likely to be the most common type of error and the most expensive to fix. Thus, proper software requirements definition is of the utmost importance.

Team Skills

To properly manage software requirements, the development team must possess the following six critical skills:

Team Skill 1, Analyzing The Problem. Developers must have a full understanding of the user's environment, the problem domain. This requires involvement of all stakeholders. Stakeholders can be the users, who are ultimately the soldiers; the customers, Army Training and Doctrine Command (TRADOC) System Managers (TSMs) who represent the user; materiel developers; and regulatory overseers, such as the Department of the Army and DOD. The Army is skilled in analyzing the problem and aware of its importance. In fact, DoD 5000.2-R states that contractors must be chosen partly on their appropriate domain experience. Unfortunately, because TSMs represent the user and are considered subject matter experts (SMEs), they are often mistaken for the actual user. The concern with this is that many SMEs are not aware of the real problem. Thus, they don't address the requirements that the actual user needs.

Team Skill 2, Defining The System. User needs and the problem domain must be defined in a vision document, which is the single most important document in a software project. The vision document captures user needs, system features, and other common project requirements. It is a living document. It is not the vision provided by the TRADOC commander, which is a very high level abstraction, documented as future operational capabilities or in mission needs statements. The vision document is closely related to the operational requirements document (ORD). But what DOD calls requirements is really a definition of system features, not functional requirements. These features can be considered the system's nonfunctional requirements, those that deal with quality of service (i.e., reliability, availability, and maintainability).

Here, the Army is a master. The work involved up to writing an ORD is vast; the work involved in writing the ORD is epic. The Army spends a great deal of time ensuring the right choice to fulfill a need is a materiel one, and then we properly assign the monumental task of building it to a PM.

Team Skill 3, Understanding User Needs. This skill consists of several techniques and subskills necessary to elicit the proper requirements from the user. The developer must include all stakeholders to gain the understanding of the problem domain. The techniques are pretty straightforward: interviewing, workshops,

brainstorming, storyboarding, role-playing, prototyping, and applying use cases—a modern approach to software development. There are some great techniques, but the Army does not benefit from all of them because there is little or no training.

Team Skill 4, Managing Scope. This involves staying within budget and schedule with reasonable flexibility. Managing the scope properly means keeping requirements in perspective. For example, the Army needs to determine if the feature is really necessary or if it is just nice to have. Once a requirements baseline is established, the PM must make tough decisions to keep requirements "creep" in check. As users and customers begin to understand the solution domain (i.e., what is possible and what is available), they are going to want more.

Another important aspect of this skill is choosing the right developmental model: waterfall, spiral, or iterative. Each has value but must be applied under the right circumstances. The waterfall model (Figure 1) is normally used when a customer must have a full working version on the first drop. The drawback is that requirements must be known upfront before any work begins.

The spiral model (Figure 2) works best when time is not of the essence and there isn't a clear understanding of the requirements. It becomes a technique to help flush out the requirements by establishing baseline requirements, analyzing developmental risk, and building prototypes. Users and customers examine the prototypes and the process starts over. The drawback is that the customer doesn't receive a working product very quickly and, quite often, wants the prototype, which is not fully functional.

The iterative model (Figure 3) can be considered the best of both worlds. It employs the benefits of both to achieve a fully functional product for earlier release to the customer. The drawback here is the user only gets a subset of the required features at each release.

Team Skill 5, Refining The System Definition. This skill involves removing ambiguity in each domain. Simply documenting the definition of the domain will create as many interpretations as there are readers. Methods must be employed to specify the requirements in such a manner so there is only one interpretation. Some methods to accomplish this are through specification languages, such as the Vienna Development Method (VDM) and Z (pronounced, Zed), through the Unified Modeling Language, pseudocode, finite state machines, and others. The features found in the ORD are ambiguous requirements. They get refined in the User's Functional Description (UFD) and the Software Requirements Specification (SRS).

The Army also does a pretty good job in this area. The UFD is the combat developer's first stab at specifying requirements and providing additional constraints, whereas the SRS is the materiel developer's first stab at alleviating ambiguity.

Team Skill 6, Building The Right System. This skill includes a multitude of techniques to keep the project on schedule and to release a product that pleases the customer. Such techniques include verifying requirements versus validating operation, requirements traceability, configuration management (CM), testing, and return on investment. A product that pleases the customer may not necessarily be the product the customer requested. It is important to provide customers a product that meets their needs; however, what they ask for and what they actually get may be two different things.

The Jury Says

To validate some of our theories, we interviewed four separate PMs and/or project leaders working in the areas of software development, software acquisition, or software systems development. The interviews were conducted using a questionnaire. All PMs we interviewed stated that the users of their system were soldiers (warfighters). They also stated that

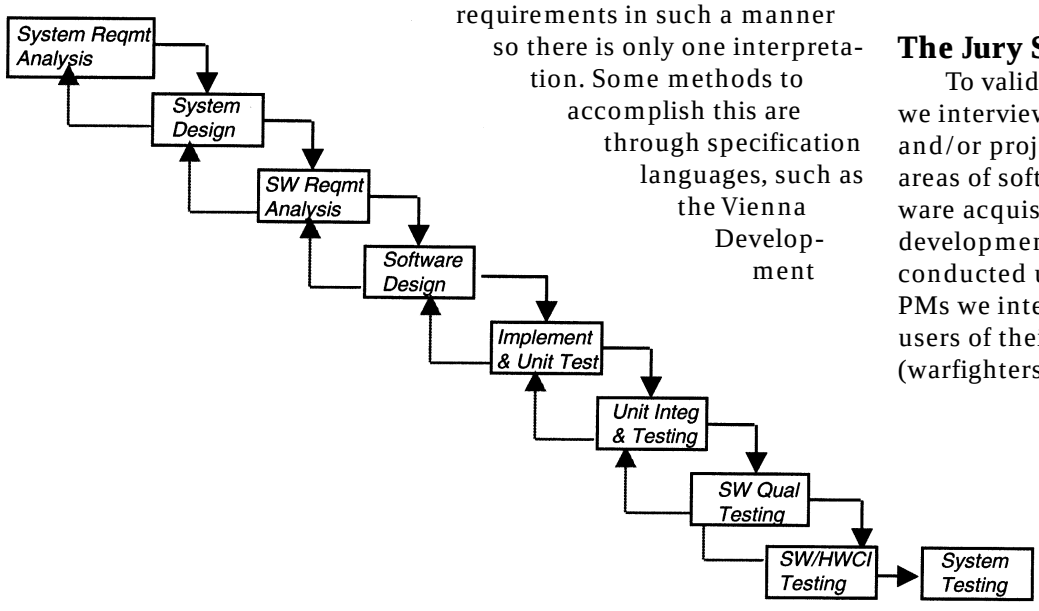

Figure 1. *Waterfall model*

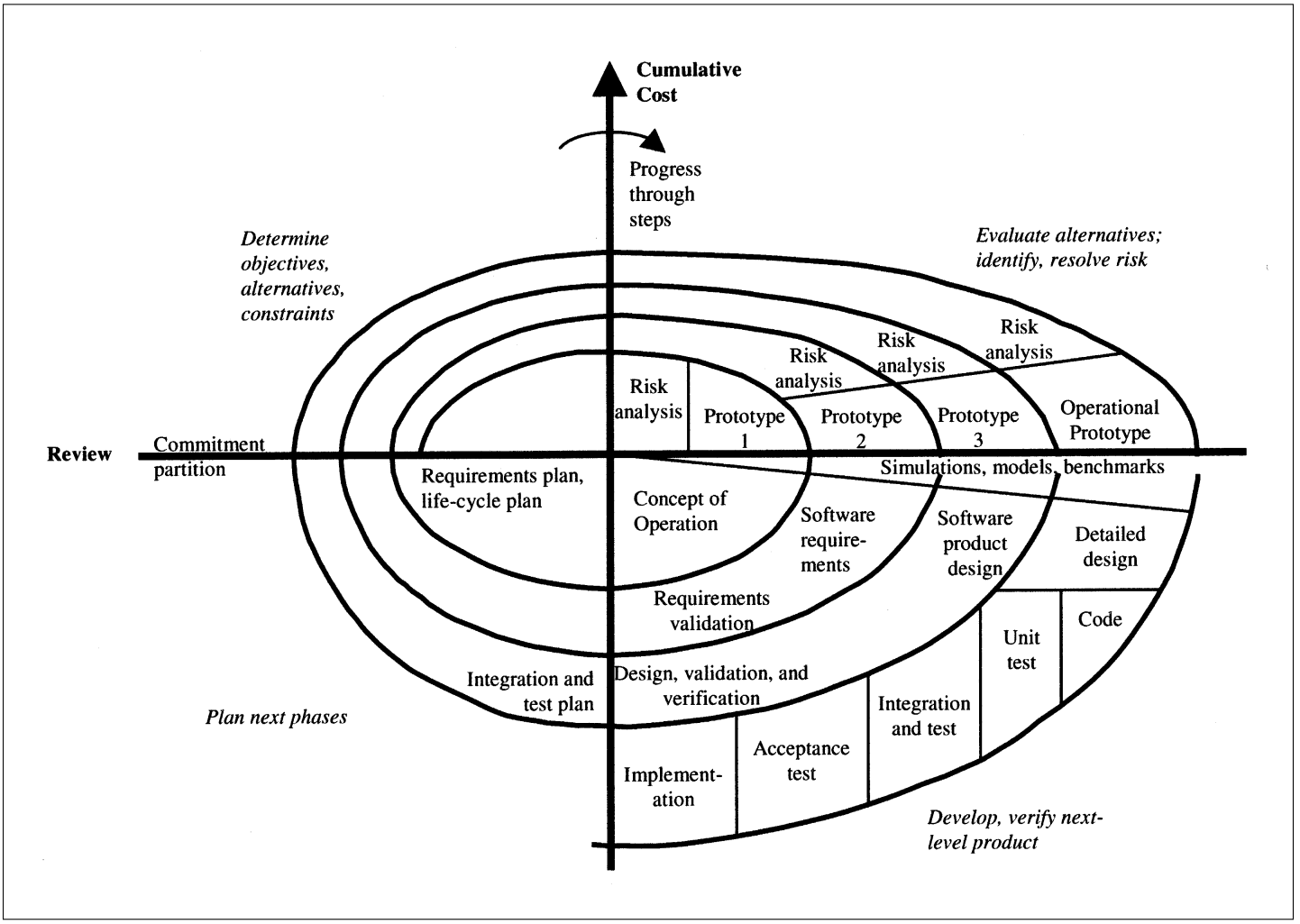

Figure 2.
Spiral model

their customers were the TSMs, functional proponents (schoolhouses), or battlefield functional area (BFA) systems—further identified as the primary stakeholder. The primary stakeholder provided them with the ORD and approved the delivered functionality of their system and the system requirements. Working groups, attended by the PM, primary stakeholder, and contractors, initially discussed the preliminary design of the system in brainstorming sessions. The SRS began to take form during these sessions and was given to the contractor for initial development. Based on input from the PM, the primary stakeholder established the priority of requirements in the SRS. The group met periodically to review requirements and conduct critical design reviews to see how well the proposed system was meeting those requirements.

All PMs agreed that the primary stakeholder did not fully represent user views. They commented that they were understaffed and not trained to properly elicit user requirements. In reality, most PMs were gathering requirements from users when they were training or testing them on their system. The PMs then needed to take these requirements back to the primary stakeholder for approval. Sometimes this was an easy undertaking, but most of the time, it required the PM to do a lot of selling of user-provided requirements.

All PMs identified contractors as their developers. Once the requirements were identified, it was up to the contractor to build the system. The contractor determined the amount of risk associated with system development and negotiated the baseline with the PM. The contractors organized, verified, and traced system requirements to ensure that the requirements were met. Some

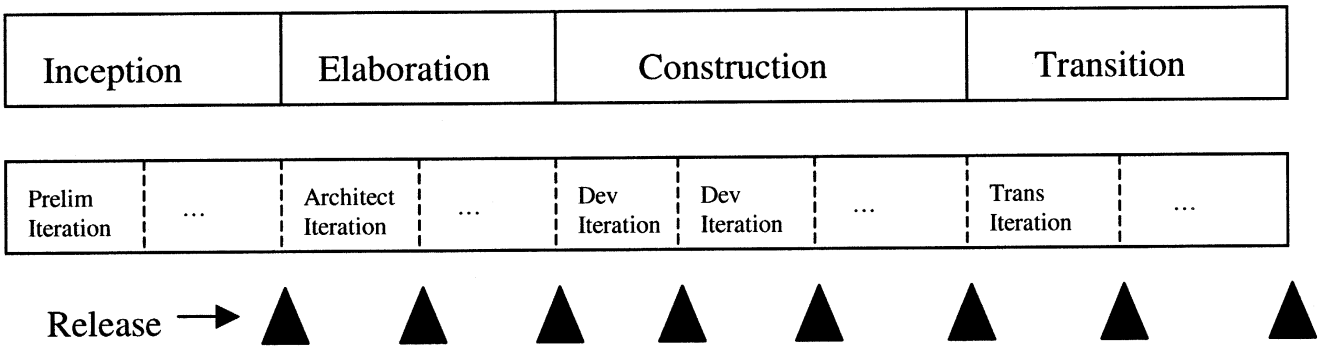

Figure 3.
Iterative model

PMs were unfamiliar with the contractors' CM methods, and others used various CM tools of their own to control and track the baseline. One PM office actually used an independent contractor to maintain CM.

In developing requirements, the PMs also had to contend with the regulatory overseers. All PMs identified the Office of the Assistant Secretary of the Army for Acquisition, Logistics and Technology and other staff agencies as overseers. These organizations provided specific guidance for the development of software systems. They did not send representatives to any of the working groups, but they were the review authority and provided the final approval for the system under development.

Most PMs stated that they used the spiral or iterative method in their development approach, but couldn't necessarily explain those models. It was determined that the costly method of test, fix, and test again was being used. Users were brought into the development process during training and testing. Comments were then provided to the PM, who went back to the primary stakeholder for approval, and the process started all over.

Conclusion

Users are not brought into the requirements analysis process; thus, they have no input until the system is built. This causes requirement errors that could be avoided by including the user earlier in the process. Changes after a system is built cost upwards of 50 times more than changes early during requirement analyses. Involving the user earlier in the analysis process would allow the developer to build to actual requirements and reduce feature creep and the "yes, buts." It is also important that workgroups involve all stakeholders in a productive session to help define the problem and elicit actual requirements. There are techniques to do this in our software acquisition process but they are not used. These techniques need to be used, in addition to training through practical exercises. Finally, our acquisition professionals need more education specifically on software engineering principles. Far too often there is too much reliance on the contractors' word with no understanding of software development.

MAJ JOSEPH P. DUPONT was a graduate student at the Naval Postgraduate School working toward an M.S. in software engineering when this article was written. He formerly worked in the Program Executive Office for Command, Control and Communications Systems and as an Assistant Program Manager in the Office of Warfighter Information Network-Terrestrial. He is currently attending the Command and General Staff Officers Course at Fort Leavenworth, KS. He holds a B.S. in electrical engineering from the University of New Hampshire and can be reached at ***joseph.dupont@us.army.mil****.*

MAJ ROBERT W. CUMMINS JR. is a graduate student at the Naval Postgraduate School, working toward an M.S. in software engineering. He formerly worked in the Office of the Deputy Chief of Staff, Information Management, U.S. Army Test and Evaluation Command. Cummins holds a B.A. in political science from Salisbury University, Salisbury, MD, and can be reached at ***robert.cumminsjr@us.army.mil****.*